第VI部

応用編

第13章　信号処理

13.1　フーリエ変換

フーリエ変換は連続変数による積分変換です．実際の多くの場合には，離散的変数値を用いた有限項の和に置きかえる必要があります．したがって，実際には離散変数のフーリエ変換，あるいは離散変数のフーリエ級数展開が必要です．

13.1.1　連続変数を離散変数に（離散フーリエ変換）

関数 $f(x)$ は区分的に滑らか（いくつかの孤立した点以外では滑らか）で連続であり，$[0, 2a]$ を基本の区間とし，周期 $2a$ の周期関数としましょう．この様な周期関数の複素フーリエ級数展開は

$$\begin{aligned} f(x) &= \sum_{n=-\infty}^{\infty} c_n e^{i\frac{n\pi x}{a}} \quad , \\ c_n &= \frac{1}{2a}\int_{-a}^{a} f(x) e^{-i\frac{n\pi x}{a}} dx \end{aligned}$$

と表されます．
区間 $[0, 2a]$ を

$$\begin{aligned} 0 &= x_0 < x_1 < x_2 < \ldots < x_{N-1} < x_N = 2a, \\ x_k &= \frac{2ak}{N} \quad (k = 0, 1, \ldots, N-1) \ , \quad \delta k = \frac{2a}{N} \end{aligned}$$

のように N 等分して，分点上の和によって c_n の積分を置き換えます：

$$c_n = \frac{1}{N}\sum_{k=0}^{N-1} f(\frac{2ak}{N}) e^{-i\frac{2nk\pi}{N}} \quad .$$

あるいはこれを

$$\begin{aligned} \omega &= \exp\frac{2\pi i}{N}, \\ f_k &= f(x_k), k = 0, 1, 2 \ldots, N-1 \\ c_n &= \frac{1}{N}\sum_{k=0}^{N-1} f_k(\bar{\omega})^{nk}, n = 0, 1, 2, \ldots, N-1 \end{aligned}$$

と書きかえます．$\{f_0, f_1, \ldots f_{N-1}\}$ から $\{c_0, c_1, \ldots, c_{N-1}\}$ への変換を **離散フーリエ変換** と呼びます．この逆変換 (**離散フーリエ逆変換**) は

$$f_k = \sum_{n=0}^{N-1} c_n \omega^{nk} \qquad (k = 0, 1, 2, \ldots, N-1)$$

となります．ω および c_n の定義により

$$\sum_{n=0}^{N-1} c_n \omega^{nk} = \frac{1}{N}\sum_{k'=0}^{N-1} f_{k'} \sum_{n=0}^{N-1} \omega^{n(k-k')} = \frac{1}{N}\sum_{\substack{k'=0 \\ (k'\neq k)}}^{N-1} f_{k'} \frac{1-\omega^{N(k-k')}}{1-\omega^{(k-k')}} + f_k$$

です．ω は 1 の N 乗根 ($\omega^N = 1$) ですから右辺第 1 項は 0 となります．これで $f_k = \cdots$ の式を直接たしかめることができました．
離散フーリエ (逆) 変換は直接この式に従って計算すると，n と k はともに 0 から $N-1$ まで動きますから，N^2 回に比例する乗・加算が必要です (正しくは乗算 N^2 回、加算 $N(N-2)$ 回)．このため N の増加にともなう計算の手間の増加は急激で，N が 300 程度になると実用上無視できない問題となります．効率よく計算を行うという点から，高速フーリエ変換 (FFT=Fast Fourier Transform) という重要かつ有効な計算アルゴリズムが発明されました．
MATLAB では離散フーリエ変換は，高速フーリエ変換を用いて行います．に具体例を示しましょう．

13.1.2 高速フーリエ変換：MATLAB を用いて

$Y = \mathrm{fft}(X)$ は高速フーリエ変換 (FFT) アルゴリズムを使用して，X の離散フーリエ変換 (DFT) を計算します．
ここでは，信号をフーリエ成分に分解してみましょう（図 13.1）．

```
Fs = 1000;          % Sampling frequency
T = 1/Fs;           % Sampling period
L = 1500;           % Length of signal
t = (0:L-1)*T;      % Time
Signal = 0.5*sin(2*pi*50*t) + 1.0*sin(2*pi*120*t);
Noise  = 2.0*randn(size(t)); X = Signal+ Noise;
subplot(2,1,1);plot(1000*t(1:50),Signal(1:50)) ; hold on
subplot(2,1,1);plot(1000*t(1:50),X(1:50))
title('Signal + Random Noise')
xlabel('t');ylabel('X(t)')
%
Y = fft(X);
P2 = abs(Y/L);P1 = P2(1:L/2+1);
f = Fs*(0:(L/2))/L;
subplot(2,1,2);plot(f,P1)
title('One-Sided Power Spectrum')
xlabel('f');ylabel('|P1(f)|')
```

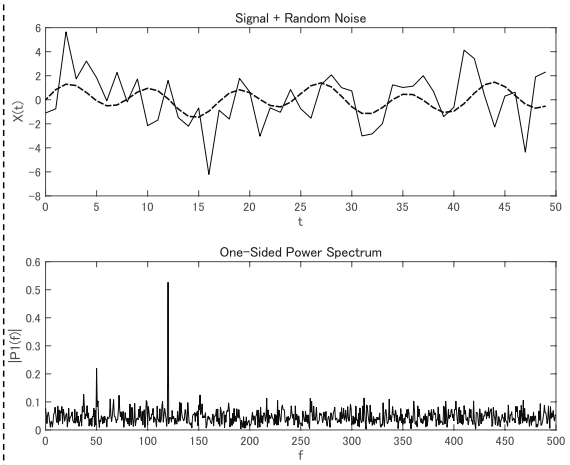


図 13.1: 信号（Signal，点線）に白色ノイズ（Noise）を重ねた実時間信号スペクトル（X，実線）（右上）とその FFT による周波数スペクトル（右下）.

- 2 つのサイン関数を重ねた信号（Signal）と白色ノイズ（Noise）を生成．この 2 つを重ねて実時間信号スペクトル（X）を作成し，これをサンプルとする（図右上）.
- Y=fft(X) により実時間信号 X から，FFT により周波数スペクトル Y を計算する（図右下）.
- 周波数スペクトル Y をプロットした右下図には，一様なノイズの中の $f = 50,\ 120$ の所に鋭いピークが見え，その強度も入力強度の相対的強さ（1：2）を反映している.

13.2　ウェーブレット変換

13.2.1　ウェーブレット変換の理論

フーリエ変換は，3 角関数を重ね合わせることにより与えられた信号を再現しようというものでした．3 角関数の性質は理解しやすく，またそれが持つ直交性，完全性も理解しやすいため，フーリエ変換は大変分かり易

いものです．しかし，それは無限に広がった周期性を持つということに起因した性質であるため，局所的に特徴を備えた信号の取り扱いは最も不得手とするところでもあります.（デルタ関数と平面波関数という両極端の波の性質は，量子力学では（実空間と運動量空間，あるいは観測時間と周波数の間の）不確定性原理として広く知られています.）

13.2.1.1 ウェーブレット変換と逆変換

このような欠点を補う目的のために，ウェーブレット（wavelet）と呼ばれる時間 t の狭い幅の中の信号波の重ね合わせで表現することが考えられています:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\Big(\frac{t-b}{a}\Big) \quad .$$

a，b を色々変えた波（波の中心をずらせたり，波の広がり方を変える）を重ね合わせることにより特徴をとらえた波を構成します.
連続なウェーブレットを用いた次の積分変換 (関数 $f(t)$ を 2 乗可積分関数として)；

$$[W_\psi f](a,b) = \int_{-\infty}^{\infty} dt \;\; \overline{\psi_{a,b}(t)}\; f(t)$$

を連続ウェーブレット変換と呼びます．$[W_\psi f](a,b)$ は a, b の有界連続関数となります.
選んだウェーブレット $\psi(t)$ がある条件（許容条件（admissibility condition）という）を満たすなら，次の逆変換が成り立ちます：

$$f(t) = C_\psi^{-1} \int_{-\infty}^{\infty} da \int_{-\infty}^{\infty} db \;\; a^{-2}\, [W_\psi f](a,b)\, \psi_{a,b}(t) \quad .$$

係数 C_ψ は $\psi(t)$ のフーリエ変換の適当な積分であり，また許容条件はこれが有限の値であることです．ウェーブレット変換は高速フーリエ変換を用いて計算できます．詳細は他の教科書（例えば「応用のためのウェーブレット」（日本応用数理学会監修，山田・萬代・芦野著，共立出版 2016 年)）を参照してください.

13.2.1.2　代表的なウェーブレット

代表的なウェーブレット（マザーウェーブレット）として，次のようなものが用いられます：

(1)　Real Shannon wavelet (sinc wavelet)

$$\psi(t) = 2\ \mathrm{sinc}\ (2t) - \mathrm{sinc}\ (t) = \frac{\sin(2\pi t) - \sin(\pi t)}{\pi t}$$

(2)　Molret wavelet　$\psi(t) = \pi^{-1/4} \exp(-\frac{t^2}{2}) \cos\left(\pi t \sqrt{\frac{2}{\log 2}}\right)$

(3)　Mexican hat wavelet　$\psi(t) = \frac{2}{\sqrt{3}\pi^{1/4}}(1-t^2)\ \exp\left(-\frac{t^2}{2}\right)$

13.2.1.3　直交基底による（離散）ウェーブレット展開

ウェーブレット $\{\psi_{j,k}\}_{j,k\in\mathbb{Z}}$ が 2 乗可積分関数系の正規直交基底となるなら，任意の関数 $f(t)$ がこれにより展開できます：

$$f(t) = \sum_j \sum_k c_{j,k}\psi_{j,k}(t)\ .$$

展開係数 $c_{j,k}$ を詳細係数と呼びます.

13.2.2　MATLAB によるウェーブレット応用：ノイズ除去とデータ圧縮

MATLAB でウェーブレットを用いるには，Wavelet Toolbox が必要です. 短時間の信号の解析のみならず，色々な局面でウェーブレット変換が使われています.

13.2.2.1　ノイズ除去

ノイズは，信号をウェーブレットに分解した後の高周波数成分に含まれ，意味のある信号情報はほとんどありません.（離散）ウェーブレット展開の展開係数（詳細係数）数 $c_{j,k}$ を，どのくらいの大きさ（閾値）以下ならどのように変更するか（ハードまたはソフト）が，一般に選択できます. 閾値は高周波数領域では高い値に、低周波数領域では低い値に設定するのが普通のようです.
図 13.2に簡単な例を示しましょう.

```
load noisdopp
denoise = wdenoise(noisdopp);
subplot(2,1,1);plot(noisdopp);
title('Original Signal')
hold on
subplot(2,1,2);plot(denoise)
title('Denoise Signal')
```

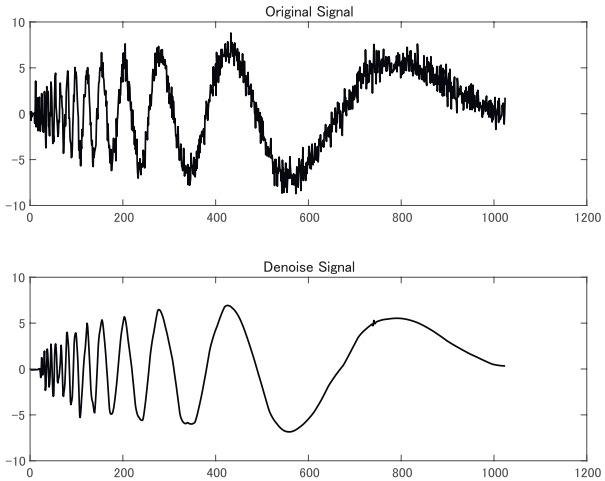


図 13.2: ウェーブレットののった信号（右上）とウェーブレット変換を用いたノイズ除去後の信号（右下）.

- Wavelet Toolbox に用意されている 1 次元信号強度の入った 1×1024 データ noisdopp (noisy Doppler signal) を Load し使用.
- ウェーブレットのノイズ除去プログラム wdenoise(x) を使用．wdenoise(x, level) とすれば閾値指定も可能．level に関してはマニュアル参照のこと.

13.2.2.2 データ圧縮

写真やイラストの画像ファイルとして用いられている GIF. JPEG, JPEG 2000 などは画像の圧縮形式を意味し，それぞれの方式で作られた画像ファイルには拡張子 .gif，.jpg あるいは .jp2 が付きます．これらのファイル形式ではデータ圧縮の際に，色数を落とす方法（GIF) あるいは色調の変化のデータを落とす方法（JPEG，JPEG2000）方式を採用しています．JPEG と JPEG 2000 では，その色を周波数成分に分解しその内の高周波成分を落とすことにより，データ圧縮を行います．JPEG および JPEG2000 では，入力画像に対する周波数変換に，それぞれ，離散フーリエ変換あるいは離散ウェーブレット変換を行います．その他にも両方式に大きな相違点もあります.

ここでは離散ウェーブレット変換を用いたデータ圧縮を，MATLAB を用いて行ってみましょう．データ圧縮は $1/2^N$ 単位で高周波成分を平滑化していきます．高周波成分を落とすため，データ圧縮を大きく進めると像の明瞭さが失われます．以下手順を追っていきます．

- **Step1** Toolboxで提供されている画像 'woman' を利用．自分の JPEG2000 ファイルを利用するなら imread('⋯.jp2') とする．

$$\begin{aligned}&\text{load woman;}\\&\text{image(X)}\\&\text{title}(\prime\text{Original Image0}\prime)\\&\text{colormap(map)}\\&\text{size(X)}\end{aligned}$$

  - load した段階で（名前は指示してないが），ファイル map（255 × 3，倍精度）と X （256 × 256，倍精度）ができている（原図は図 13.3）．
  - image(X) は、配列 X 内のデータをイメージとして表示する．
  - colormap(map) は画像ファイル X のカラーマップを， map で指定されたカラーマップに設定．
  - size(X) でデータ X の情報（256 × 256）を確認．

- **Step2**　データ分解のプロセス．

$$\begin{aligned}&\text{wv} =\prime \text{db1}\prime;\\&[\text{cfs}, \text{inds}] = \text{wavedec2}(\text{X}, 3, \text{wv});\end{aligned}$$

  wavedec2(X,3,wv) は ウェーブレット wv（上の指定では，Daubechies ファミリ の db1-ウェーブレット）を使用して行列 X の レベル 3 の 2 次元ウェーブレット分解を行う．

- **Step3**　分解データ cfs からレベル 2（64 × 64），およびレベル 3（32 × 32）のデータを抽出．レベル 2 のスクリプトのみ，図 13.4に

図 13.3: 画像データ圧縮のためのサンプル（256 × 256).

は両方の画像を示す.

```
cfs2 = appcoef2(cfs, inds, wv, 2);
figure;
imagesc(cfs2)
colormap('gray')
title('Level2ApproximationCoefficients')
size(cfs2)
```

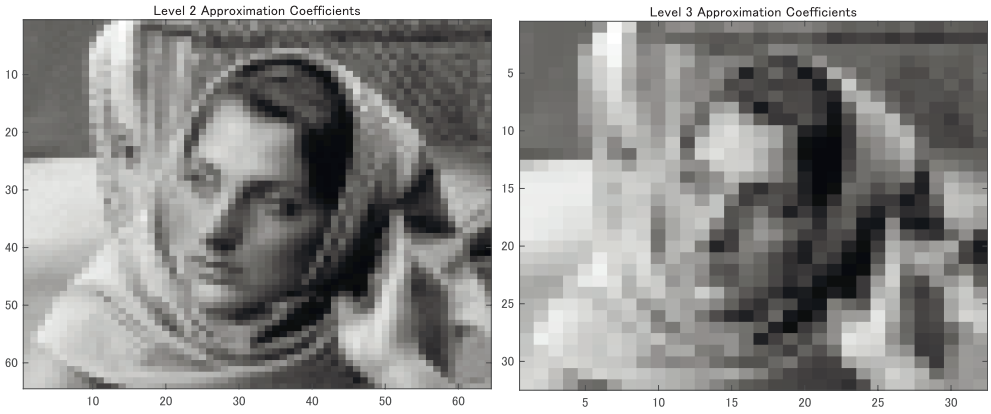


図 13.4: ウェーブレットによりデータ圧縮された画像.（左）レベル 2（64×64）と（右）レベル 3（32×32）.

第14章　行列の特異値分解を用いた低ランク近似と画像圧縮

デジタルな画像について，その処理に特異値分解を応用してファイル容量を圧縮することができます.

14.1　画像の行列表現と低ランク近似

画像データは$n \times m$ピクセルの点に3色に分けて（サイズ$n \times m \times 3$) 収められます.
簡単のために画像を白黒画像とし，そのデータをどの程度圧縮できるか考えてみましょう．画像データを2次元行列に見立て，その特徴を少ない主要な要素で再現しようというものです．そのために**行列の特異値分解を用いた低ランク近似**を採用します.

もとの画像が白黒で$n \times m$行列Iで与えられているとします．ピクセル数は変えないでおきます．データ量は

$$n \times m$$

です．実際にはこのファイルに8bitsデータを載せて，各点の明度を2進数を用い256段階（$2^8 = 256$）で表します.

この行列を特異値分解

$$I = U * S * V^T$$

するのです．特異値分解のために数値計算をします．そのためには8bitsデータを実数型に直さなくてはいけません.

必要な情報データ量は $P = \min(n, m)$ として，

$$P \times (1 + n + m)$$

です．$\min(n, m)$ は，n か m のいずれか小さい方の値を意味します．
このままでは最初のデータ量 $n \times m$ より大きくなります．しかし画像を認識するにはこのようなデータ全部が必要ではありません．P をどの程度小さくできるかがポイントです．P を n, m の 10 分の 1 にすることができれば，全体として数分の 1 までデータを圧縮しても元のデータが持つ情報量を落とさない，あるいは元の画像が持つ特徴を，特異値という形でとらえることができた，ということになります．

14.2　MATLAB を用いた画像圧縮

利用する特異値が k 個であるとして，k をどの程度まで小さくできるか，データ量

$$k \times (1 + n + m)$$

をどこまで小さくできるかがポイントとなります．実際に MATLAB を用いてやってみましょう．

Step1 指定された画像ファイル YasudaP.jpg を行列 img として読み込みます：

```
>> img = imread('YasudaP.jpg');
```

この中身がどのようなファイルか，whos コマンドを用いて確認します．

```
>>  whos  img
  Name          Size              Bytes    Class    Attributes
  img     2515 × 2525 × 3       19051125   uint8
```

画像ファイル YasudaP.jpg は　img(2515, 2525, 3) ファイルとして読み込まれました．unit8 はデータ形式が 符号なし整数 (unsigned integer) 8bits ファイルであることを示しています．

Step2-1 カラー画像は RGB 画像といい，3 枚の二次元画像から成ります．R(red)，G(Green)，B(blue) が，行列の 3 番目の引数の 1 ～ 3 に対応します．

$$>> \quad \text{redImg} = \text{img}(:,:,1);$$
$$>> \quad \text{greenImg} = \text{img}(:,:,2);$$
$$>> \quad \text{blueImg} = \text{img}(:,:,3);$$

とすれば，RGB の 3 要素を分離して其々が白黒画像に変換されます．

これらの画像ファイルを開くには

$$>> \text{imtool(redImg)}$$

とするか，

$$\text{mwrite(redImg,}' \text{redImg.jpg}');$$

として画像ファイルに出力するかの 2 つの方法があります．

Step2-2 カラー画像を一度に白黒画像に変換するためには rgb2gray という関数が用意されています．

$$>> \quad \text{grayImg} = \text{rgb2gray(img)};$$

この関数 rgb2gray では重み付加算

$$\text{gray} = 0.2989\text{R} + 0.5870\text{G} + 0.1140\text{B}$$

を用いています．

- このファイル grayImg は少し大きすぎますので，ファイルサイズを小さくしておきましょう．そのために 画像ファイルを小さくするための関数 imresize を使います．

$$>> \text{grayImg0} = \text{imresize(grayImg, 0.2)};$$

これでファイルのピクセル数が 0.2 倍になります：

```
>>  whos grayImg0
  Name        Size          Bytes    Class    Attributes
  grayImg0    503 × 505     254015   uint8
```

$254015 = 503 \times 505$. 数値は Bytes 単位です（2^8 : 8bits=1Bytes)

Step3 これから行列の特異値分解を行いますが，整数クラス uint8 のままでは数値演算に不向きです．そのため行列のデータを関数 double を用いて 64bits 倍精度浮動小数点数に変換します．

```
>> grayImggray0 = double(grayImg0);
>> whos  grayImggray0
```

Name	Size	Bytes	Class	Attributes
grayImggray0	503×505	2032120	double	

$2032120 = 503 \times 505 \times 8$：64bits=8Bytes

- これで特異値分解の準備が整いました．

```
>> [U, S, V] = svds(grayImggray0, 50);
```

ここでのパラメータ 50 は，特異値の大きな方から 50 個で分解するということを指示します．

こうして 50 個の特異値で行列を再成します．

```
>> gray50 = U * S * V';
```

U, S, V の大きさを whos コマンドを用いて確認してみます：

Name	Size	Bytes	Class	Attributes
gray50	503×505	2032120	double	
U	503×50	201200	double	
V	505×50	202000	double	
S	50×50	20000	double	

S は対角行列ですから 50×50 は不要で 50 だけで十分です．こうしてデータ量が大きく圧縮されました．

$$503 \times 505 \to 50 \times (1 + 503 + 505)$$

$k = 50$ を色々変えて，画像の再現性を確認してみてください．実際には k の値をもっと小さくしても何が映っているか充分識別できるでしょう．

Step4 gray50 は double のデータ型行列です．画像ファイルとするために uint8 型に変換します．

```
>> igray50 = uint8(gray50);
```

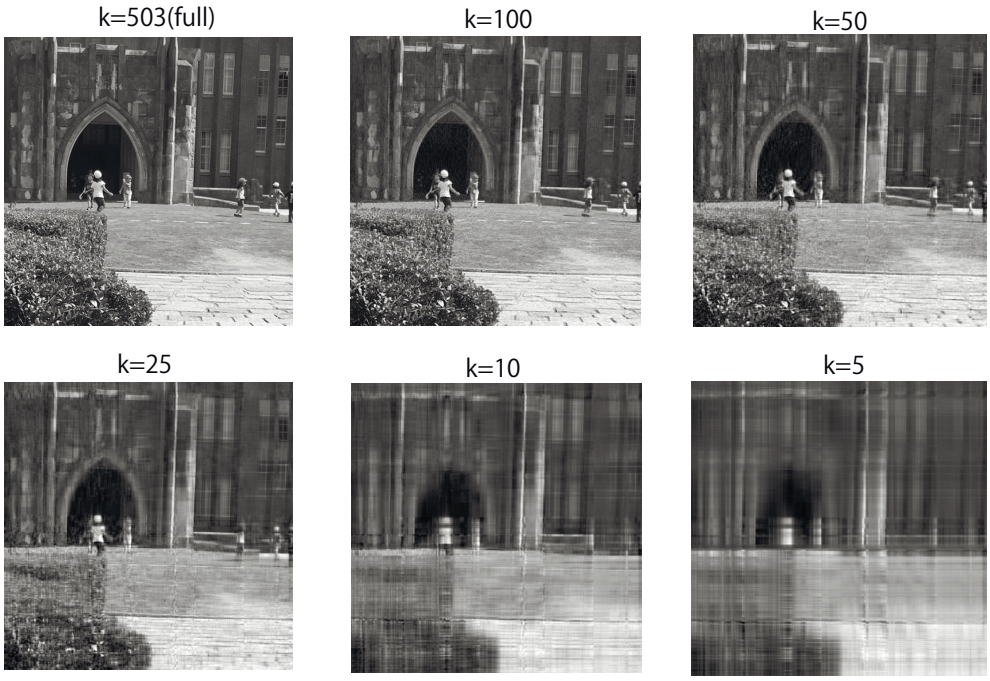


図 14.1: 特異値を用いた画像の低ランク近似（安田講堂前で遊ぶ幼児たち）．大きい方から k 個の特異値（固有値）を用いた．

Step5 最後は結果を画像ファイルとして保存します．

$$\text{imwrite}(\text{igray50},'\,\text{gray50.jpg}');$$
$$\text{imwrite}(\text{igray50},'\,\text{gray50.png}');$$

これらの結果を図 14.1に示します．特異値の 10 分の 1 程度（$k \simeq 50$）で充分に元の画像が再現されています．

第15章　シミュレーション

15.1　シミュレーションとは

実際のシステムを動かしてその動きを確かめることが難しい，危険だ，手間がかかり過ぎるため条件を種々変えることができないなどの場合，あるいは，実際の現象がゆっくりしているため限られた時間内に知見を得ることができない，逆に非常に高速な現象であったりシステムの内部の現象であるため観測不能である，という場合など，興味があり重要であるにもかかわらず観測できない現象が数多くあります．また現象をコントロールするパラメータを人為的に変えることができず，現象の知見を充分に得ることが難しいこともしばしばです．このような場合，シミュレーション（simulation）という手法によってシステムの特性や実際の現象の実際が調べられます．
実物大の模型あるいは実物を縮小した模型を作って風洞や水槽の中で試験するのもシミュレーションの一種です．ここでとり上げるのは，実際の現象を表す**数理モデル**で置き換え，コンピューターを用いた**数値実験**をおこなう方法です．具体的には，電気系，機械系のシステム，原子スケールでの物理・化学現象，地球スケールや宇宙スケールでの自然現象，社会現象に関して広く用いられています．

15.2　Simulink

MATLABでは，シミュレーションツールとして，Simulinkが用意されています．Simulinkは，様々な系の数学モデル（微分方程式等）を配置し観測するための，MATLABの下で動く**GUI（Graphical User Interface）**です．したがって容易にモデルの系を設定することができます．
Simulinkでは，**ブロック図ツール**とそれらのブロック図を種類ごとにまとめた**ブロックライブラリ**が入出力および機能要素として提供されています．

15.2.1 Simulink の使い方 1.

ここでは Simulink の立ち上げと簡単な使い方を紹介しましょう.

15.2.1.1 Simulink の立ち上げ

Simulink を立ち上げるには 2 つの方法があります.

方法 1 コマンドウィンドウに 「simulink」と入力.

方法 2 ホームタブから

1 ホームタブの「Simulink」アイコンをクリック

2 （あるいは）ホームタブの「シンク作成」アイコンをクリックして現れるスクロールバーから「Simulink モデル」を選択.

方法 1，方法 2 のいずれでも「Simulink スタートページ」が起動します.

15.2.1.2 Simulink の「モデルブラウザー」の起動

次に Simulink の操作を行う「モデルブラウザー」を起動します.

方法 「Simulink スタートページ」の「Simulink」から「空のモデル」を選択.

これで「モデルブラウザー」が新たに起動します.

15.2.1.3 Simulink の「ライブラリ ブラウザー」の起動と「ブロック」の選択

方法 「モデルブラウザー」の「シミュレーション タブ」にある「ライブラリブラウザー」アイコンをクリック.

以上で,「Simulink ライブラリ ブラウザー」が開きます.
この「ライブラリ ブラウザー」の中から，必要な部品を探します．ライブラリーは，機能ごとにグループ化され，ブロック毎に置かれています.「ライブラリ ブラウザー」の検索の窓を使うと効率の良い選択が可能ですが，名前で検索しますので慣れるまで少し時間がかかるかも知れません.

15.2.2　Simulink の使い方　2．

以上で準備が整いましたので，実際の Simulink 使用の手順に移りましょう．ここでは基本的な利用と RL 回路（または簡単な力学系）を例にとってみましょう．

15.2.2.1　波形のモニター

波形の発生とそのモニターの 2 つから成り立ちます．

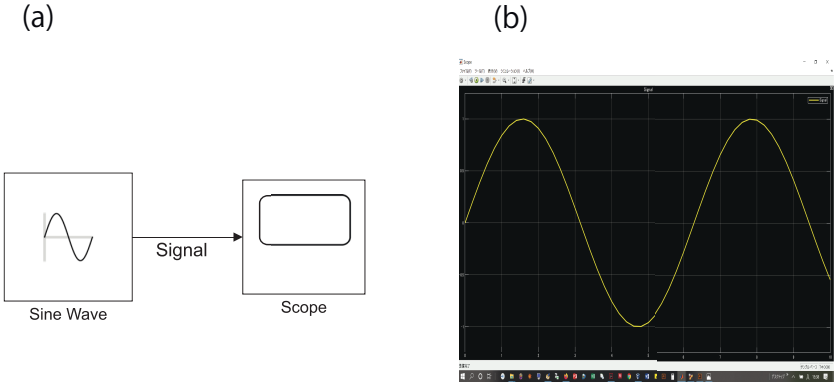


図 15.1: (a) モデルブラウザー上の，ブロックから構成されたモデル．(b) モニターに現れる信号．

Step1 Sources Blocks から「Sine Wave」を選び,「モデル ブラウザー」上にドロップする（波形の発生）．

Step2 Commonly Used Blocks から「Scope」を選び,「モデル ブラウザー」上にドロップする（信号のモニター）．

Step3 「Sine Wave」と「Scope」を**結線**する．A から B へ結線したい場合は , [Ctrl] キーを押しながら, マウスで A, B ブロックの順に左クリックする．結線上に名前を付けたいときには，描きたいところでダブルクリックして記入する．
Simulink のためのファイルを…という名前で保存すると，slx という拡張子が付き，….slx という名前が付く．

Step4 モニター信号を見るために「Scope」ブロックをダブルクリックしてグラフをオープンする．その上で，シミュレーションタブ上の実行ボタンをクリックする．

Step5 「Sine Wave」ブロックをダブルクリックすると,「ブロックパラメーター」ウィンドウが開く．ここで入力信号のパラメータを選択できる．

図 15.1にこれらを示します.

15.2.2.2 2 階常微分方程式系

2 階常微分方程式で表されるシステムは沢山あります．ここでは 1 質点振動系を考えます．問題にする微分方程式は

$$m\frac{d^2x}{dt^2} + \gamma\frac{dx}{dt} + kx = f(t)$$

です．x は質点の時間 t に依存する変位，m は質点の質量，γ は (減衰) 抵抗定数, k はばね定数, $f(t)$ は時間に依存する外力です.
上の式を

$$\frac{d^2x}{dt^2} = \frac{1}{m}\left[f(t) - \gamma\frac{dx}{dt} - kx\right]$$

と書き直しましょう．このようにすると,「外力，粘性抵抗，ばねによる力を足し合わさせて $1/m$ 倍すれば，x の 2 階微分（加速度）となる」と読めます．そしてそのようなブロック線図を書けばよいということになります.

Step1 「Sum」ブロックを探しモデルブラウザーにドロップ.

Step2 「Sum」ブロックをダブルクリックし，ブロックパラメータ ウィンドウを開き，符号リストに「＋－－」と入力する．これにより Sum block の入力の数と符号を指定．＋を $f(x)$ に，残りの 2 つを $-\gamma\frac{dx}{dt}$ と $-kx$ に割り当てることができる.

Step3 出力全体に $1/m$ を掛けるために,「Gain」ブロックを探してモデルブラウザーにドロップ．Sum ブロックから Gain ブロックに結線.

Step4 （Gain ブロックをダブルクリックし，パラメーターウィンドウを開いて,）Gain ブロックの重みを $1/m$ に書き換える．m の大きさは後で指定．Gain ブロックの出力は $\frac{d^2x}{dt^2}$ であるので，そう書いておこう.

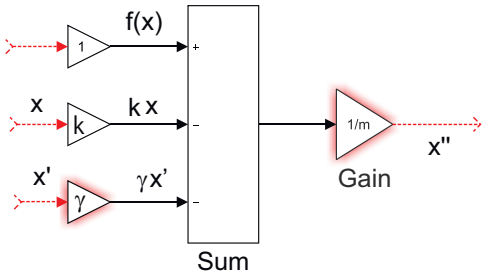


図 15.2: 振動子系モデル 1 .

その様な様子は図 15.2に示されています．次に，この力学系ダイアグラムを完成し、必要な量に対するモニターを置きます．

まず x' および x が必要です．そのために x'' を 1 回，さらにもう 1 回積分します．

Step5 積分のための「Integrator」ブロックを探しモデルブラウザーにドロップ．

Step6 必要なところを結線して全体の系を完成．

Step7 外力として適当なものを選択．ここでは，ある時刻（0）にゼロから有限の値に立ち上がり，その後一定値を保つ「Step」を外力とする．観測の必要なところに「Scope」をつなぐ（図 15.3）．

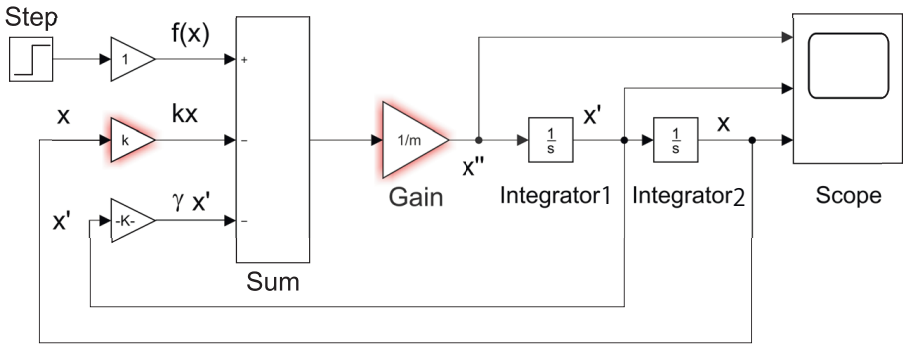


図 15.3: 振動子系モデル 2.

これで準備が完全に終わったのではありません．振動子系の定数 m, k, γ の値を決めなくてはなりません．

Step8 「Scope」をダブルクリック．

Step9 パラメータの値を設定するには 2 つの方法がある．
1. コマンドラインからの直接入力．
2. ….m ファイルで，パラメータの値を設定．そのファイルを通常の方法で走らせる．

Step10 Simulink のファイル….slx を実行する．結果は Scope に出る（図 15.4）．

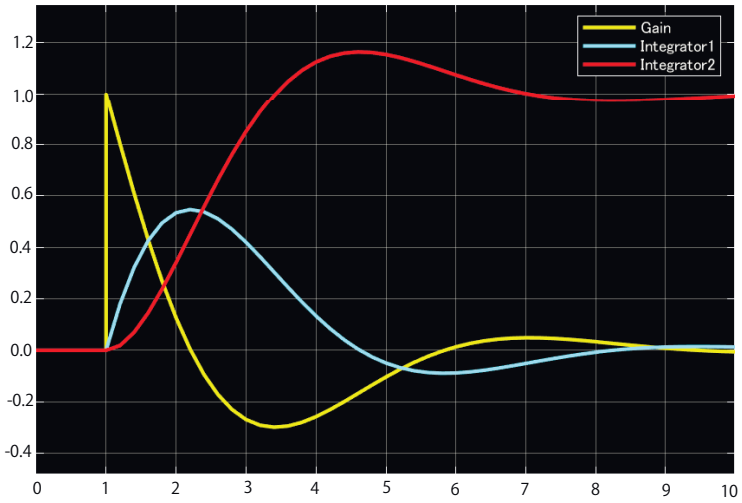


図 15.4: 振動子系モデル 3.

図 15.4で，ステップ的な外力が与えられて加速度が 0 から有限の値に立ち上がり，それに伴って測度および変位が 0 から連続的に変化すること，また粘性抵抗に従って振動が減衰する様子も観測されます．Scope において，3 つの信号が 1 つの図にプロットされていますが，それぞれの単位は違います．Scope ブロックのパラメータを変えれば 3 つのそれぞれを，異なる図に描くこともできます．

第16章　深層学習，機械学習など

16.1　人工知能，機械学習，深層学習とは

人工知能（AI=Artificial Interiggence），機械学習（Machine Learning），深層学習（Deep Learning）などの言葉が，巷では多く聞かれる時代になりました．具体的に何がどのように違うのか判然としないまま，呪文のように口にしているように思われる場面も多く見かけるようになりました．はじめに人工知能，機械学習，深層学習というキーワードをきちんと区別しておきましょう．全体の概念構成は

$$\text{人工知能} \supset \text{機械学習} \supset \text{深層学習}$$

となります．

- 人工知能（AI) という概念は 1950 年代に形成されたもので,「人間にしかできなかったような高度に知的な作業や判断をコンピュータを中心とする人工的なシステムにより行えるようにしたもの」を意味する（IT 用語辞典）．具体的応用分野として，画像認識システム，音声認識システム，自然言語処理システム，自動運転システムなどが実現されている．

- 機械学習とは、人工知能を実現するための手法の 1 つの方法で，人間の経験による学習と同様な学習機能をコンピューターに教えるアルゴリズムをいう．機械学習では，機械的（コンピューター的）方法により，データから直接情報を ”学習” する．学習するデータが増えるにしたがい，一般に，獲得する性能が向上する．

  機械学習では分析のためのロジックを全てプログラムするのではなく，コンピュータによる学習により，大量のデータから，その特徴をグループ分けする。その獲得したロジックによって，さらに与えられたデータに対して自律的に高度の予測を行う手法が構成される．

学習の過程は「入力（データの前処理，適切な表現形式の選択）」「処理」「出力（回帰などのデータ予測）」から構成される．処理の過程には**ニューラルネットワーク**（neural network）という人間の脳を構成するニューロン（neuron）のネットワーク形成を模したアルゴリズムを用いる．

- 深層学習とは，機械学習を有効に行うための方法．最初の処理の結果をさらに入力として処理するというように，処理の過程（ニューラルネットワークの構成）を多層的・多段的に行う．これにより有効な機械学習が実現された．

以下で機械学習，ニューラルネットワークに関していくつか紹介したいと思います．多分，これらに付いて解説書を読むより，実際に即して計算を経験するのが分かり易いように思います．

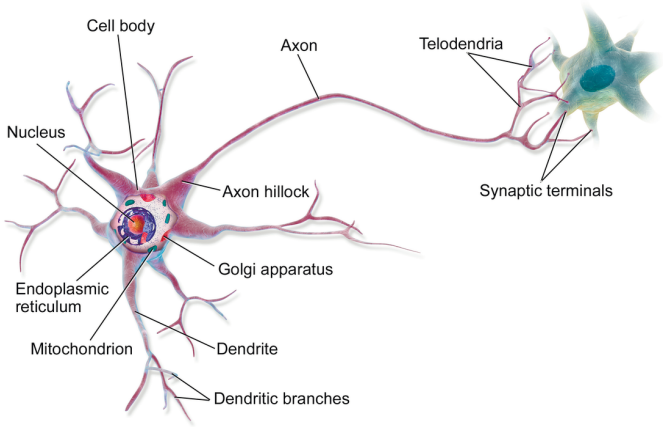


図 16.1: ニューロン．Wikipedia より（C.C. BY 3.0).

16.2 深層学習

機械学習といっても，このソフトウェアにデータを投げ込むと，知らないことも解決してくれる，というようなものはありません（多分!?）．下記項目はお互いにそれぞれがかかわりを持って機械学習を構成します．

16.2.1　機械学習

MATLAB では Statistics and Machine Learning Toolbox が用意されています．そこでは，統計および機械学習による解析とモデル化が基本となります．

16.2.1.1　機械学習のためのステップ

機械学習は以下のようなステップ（の総て，または一部）から構成され，そこでは様々な基本的数理手法が反復して使われます．
https://jp.mathworks.com/solutions/machine-learning/resources.html?s_tid=conf_addres_DA_eb の MATLAB 「機械学習入門」に多くの解説・実例があります．

- 入力データ
  - データの表現：画像データ，信号データ，文章（自然言語）
  - モデル：[観測データ] $\leftrightarrow$ [ラベル]
- 情報の抽出（データマイニング）
  - 最適化
  - 回帰分析
  - EM アルゴリズム
- 回帰（regression）と分類（classification），クラスタリング（clustering）
  - 統計手法（回帰分析）
  - 「教師あり学習」と「教師なし学習」
- ニューラルネットワーク
  - 最適化問題
  - 「教師あり学習」と「教師なし学習」
- 評価，検定
  - 訓練データと評価データ

16.2.1.2 入力データ

入力データは，音声データ，画像データ，数値データ，文章など様々な形が考えられます．例えば画像データの場合，画素単位の RGB 値や明るさあるいはその広がりなどが入力データとなります．文章の場合には，それらを単語に切り分けたものとそのつながりが入力データとなります．

16.2.1.3 「分類問題」における「教師あり学習」,「教師なし学習」

各データを表すサンプル点が特徴空間（画像なら RGB 値や明るさをそれぞれ要素とする多次元空間）に分布しているとき，このサンプル点を分類する方法は，大きく分けて二つあります．

- 「教師あり学習」の分類問題である **k 近傍法**（k-nearest neighbor algorithm, k-NN）．分類したいクラスタの特徴点からの空間内の距離によりサンプル点を分類していく．
- 「教師なし学習」の分類問題である **k 平均法**（k-means clustering）．クラスタの数を決め，最初にサンプル点に乱数によってクラスタを割り当てる．各クラスタに属するサンプル点の重心を求めクラスタの中心とする．サンプル点を近いクラスタ中心に割り当て直す．これらの作業を何回か繰り返す．

16.2.1.4 最適化

既に説明した方法により最適解を求める過程．統計処理における「推定」と「検定」，あるいは「線形計画法」と「非線形計画法」など．

16.2.1.5 潜在変数の抽出 ー EM アルゴリズム

EM アルゴリズム=expectation-maximization algorithm
サンプル点の分布 $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots \boldsymbol{x}_N\}$ に隠れた変数（潜在変数）z がある場合に，その潜在変数を最尤推定する手法の一つ．手順を（不十分ではありますが）少し具体的に示すために，いくつかの定義式および関係式を

提示します．より詳細には，例えば「機械学習」（中川祐志著，東京大学工学教程 （丸善出版））を参照してください．

$$\begin{aligned}
\text{データの集合} &: \mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots \boldsymbol{x}_N\} \\
\text{サンプル点 } \boldsymbol{x}_n \text{に対する潜在変数の値} &: z_n \\
\text{確率分布を決めるパラメタの集合} &: \boldsymbol{\theta} \\
\text{条件付確率} &: p(\boldsymbol{x}, z|\boldsymbol{\theta}) = p(z|\boldsymbol{x}, \boldsymbol{\theta}) p(\boldsymbol{x}|\boldsymbol{\theta}) \\
\text{潜在変数 } z \text{ 込みの対数尤度} &: \log L(\boldsymbol{x}, z|\boldsymbol{\theta}) = \sum_{n=1}^{N} \log p(\boldsymbol{x}_n, z_n|\boldsymbol{\theta})
\end{aligned}$$

$\boldsymbol{\theta}$ を分布 $\mathcal{D}$ から推定しよう，その場合に対数尤度を最大にするように決めよう，というのがここでの問題です．
観測変数 $\boldsymbol{x}$ と潜在変数 z の同時分布 $p(\boldsymbol{x}, z|\boldsymbol{\theta})$ が与えられています．このときの EM アルゴリズム（対数尤度関数 $\ln p(\boldsymbol{x}|\boldsymbol{\theta})$ の $\boldsymbol{\theta}$ についての最大化）は以下のように行われます．

Step1 ．　初期化．初期値 $\boldsymbol{\theta}^{(n=1)}$ を適当に決める．

Step2 ．　(E ステップ)　z の事後分布 $p(z|\boldsymbol{x}, \boldsymbol{\theta})$ を計算する．

Step3 ．　(M ステップ)　事後分布 $p(z|\boldsymbol{x}, \boldsymbol{\theta})$ による対数尤度の期待値

$$Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(n)}) = \sum_{z} q(z) \ln p(\boldsymbol{x}, z|\boldsymbol{\theta})$$

を計算する．
ただし $q(z) = p(z|\boldsymbol{x}, \boldsymbol{\theta}^{(n)})$ と選ぶ．ここは既知の情報から決まる z の関数なら何でもよいのだが，一般にこのように選ばれる．
$\boldsymbol{\theta}$ は $Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(n)})$ を最大化するように決める：

$$\boldsymbol{\theta}^{(n+1)} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(n)}) \ .$$

Step4 ．　$\boldsymbol{\theta}^{(n+1)}$ に対応する対数尤度関数 $\ln p(\boldsymbol{x}, z|\boldsymbol{\theta}^{(n+1)})$ を計算し，これが収束条件を満たしていない場合には Step2 に戻る．

一般に EM アルゴリズムがどういったこと をしているのかについての直観的理解の助けとなると考えられる．

16.2.1.6 学習による結合の強化と Overfitting,「過学習」,「過適合」

学習により上式の重み w_i の重み付けを変化させます．一般に与えられたデータにモデルを適合させることは，可能です．一方で与えられたデータ（訓練データ）に適合させすぎると，モデルの予測能力が落ちることはしばしば経験することです．このような状況を overfitting，日本語で「過学習」あるいは「過適合」といいます．これを如何に避けるかは機械学習，深層学習では重要な課題です．1 つの対応策は，モデルにある程度の柔軟性を持たせることです．その様な方法を「正則化」といいます．

16.2.1.7 クラス分類，多クラス分類（ロジスティック回帰）

パーセプトロンで 2 クラスに分類することは上の式の通り．多クラス分類は一般化（多値）．

16.2.2 ニューラルネットワーク

- ニューロン：動物の神経細胞は，細胞核のある細胞体，他の細胞からの入力を受ける樹状突起，他の細胞に出力する軸索から成り立ち，これら全体をニューロン（neuron）という．軸索の先端は他の神経細胞に接続してその間にシナプス（synapse）を形成する．ニューロンは他のニューロンからの刺激（信号）を受け取り，シナプスでそれを増減させ他のニューロンに伝達する．

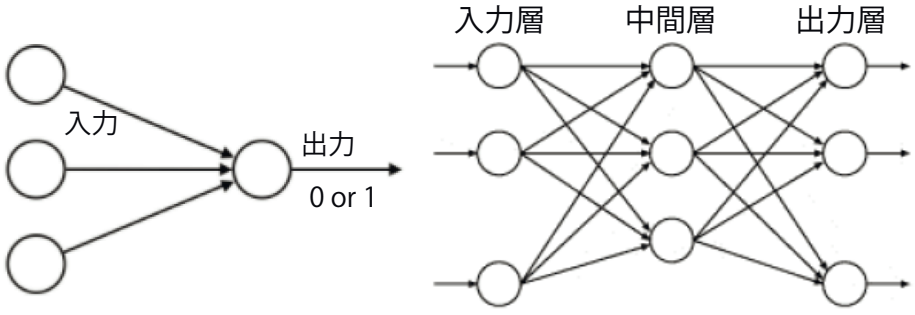


図 16.2: パーセプトロン（左）と階層型ニューラルネットワーク（右）.

- パーセプトロン ：人工ニューロン．複数の入力に対して 0 か 1 の 1 つの出力を与える回路あるいは関数（下式（単純パーセプトロン）および図 16.2左）．各入力を x_i，出力を y と書くと

$$\begin{aligned} x &= I_0 + w_1 x_1 + \cdots + w_n x_n \\ y &= h(x) \\ h(x) &= \begin{cases} 0 & (x \leq 0) \\ 1 & (x > 0) \end{cases} \end{aligned}$$

  また I_0 をバイアスと，$h(x)$ を一般に活性化関数という．活性化関数には，ここで示した階段関数のほかに，0 と 1 の間を滑らかにつなぐ種々の非線形関数が考えられている．

- ニューラルネットワーク：人工ニューロンから構成したネットワークが，学習によってシナプスの結合強度を変化させ，問題解決能力を持つようにしたモデル（図 16.2右）．

- 深層学習：多層のニューラルネットワークによる機械学習．

16.3　MATLAB に用意されている深層学習

かなりのことは Deep Learning Toolbox を用いて実行できます．

1. MATLAB は，種々の入力に対する機械学習，深層学習の多くのプログラムを提供しています．それらを動かし，また入力データを自前のものに取り換え，あるいは書き換えるなどしながら具体的に学ぶことが可能です．

2. Deep Learning Toolbox では，Simulink を用いてニューラルネットワークを作ることができるようですので，便利だと思います．

MATLAB が提供する具体的な例を見ることにしましょう．既に用意され調整されているパッケージや関数を用いますが，それらに慣れた上で自分で少しずつ変更を加えたり，新たに書き加えするのが良いでしょう．その場合に，関数のコードがどのように書かれているかを見るのには，type コマンドを使うのが便利です．

```
type file_name
```

16.3.1 例：機械学習による 5 種類の「木の実」の識別・分類

MathWorks 社提供による NutMeasurement.mlx の解説を与えることにしましょう．画像に写っているものを識別・分類する機械学習のプロセスとして，大変具体的かつ詳細に実行していますので，良い教材といえるでしょう．
以下の例は MathWorks 社に提供して頂いた機械学習を用いて画像の中の木の実を分類するものです．入力データは画像として与えられています．総てのコマンドは MATLAB で与えられていますので，マニュアルで確認してください．またプログラム中の図の出力は，一部のみここに示してあります．実際に自分で動かしてみてどのような出力が得られるのか確かめてください．

- 画像データを読込み

```
I = imread('DSC_1895cr.png');
figure; imshow(I);
```

図 16.3: 入力画像 （DSC_1895cr.png）.

- データの準備（情報の抽出=ナッツ領域の切り出し）
  - 輝度変換 （RGB 値を人の色空間を近似する CIE1976 L*a*b* 色空間 (CIELAB) に変換.

```
Ilab = rgb2lab(I);
[L, a, b] = imsplit(Ilab);
```

- エッジ検出 （強度の微分強度が大きなところを検出）.

```
[Iedge, th] = edge(L, 'canny', 0.06);
imshowpair(L,Iedge, 'montage')：二つの像を並べて比較.
```

- Morphological Closing という手法を用いて作った閉じた図形（穴埋めされた図形）の像を返す.

```
bw = imclose(Iedge, strel('disk', 3));
bwNuts = imfill(bw, 'holes');
imshow(bwNuts)
```

以上で ナッツ領域の切り出しが完了（それを確認のため imshow）.

```
figure, imshow(I .* repmat(uint8(bwNuts), [1,1,3]));
```

- 分類

- 形状で分類: Area（面積） と Eccentricity（楕円度） で「ジャイアントコーン」と小さいグループ大きいグループに分類する．次のコマンドでイメージの領域解析 アプリを開く.

```
imageRegionAnalyzer(bwNuts);
```

「stats = regionprops(BW,properties)」 はバイナリイメージファイル BW 内の各 8 連結要素 (オブジェクト) について properties で指定されるプロパティの測定値を返す．properties は多種類が用意されているので，マニュアルを参照.
Area と Eccentricity でジャイアントコーンと小さいグループ大きいグループに分類可能.

```
stats = regionprops(bwNuts, 'Area', 'Eccentricity', ···
  'Circularity', 'MaxFeretProperties', 'BoundingBox');
  areas = ［stats.Area］;
  eccentricity = ［stats.Eccentricity］;
```

ヒストグラムで確認

```
figure, h=histogram(areas, 10); title(' 面積');
```

```
figure, h=histogram(eccentricity, 10); title(' 楕円度');
```

楕円度で「ジャイアントコーン」を分類，インデックスを付与.

```
idx_GiantCorn = [stats.Eccentricity] < 0.5;
Idetect = insertObjectAnnotation(I, 'rectangle',···
  [cat(1,stats(idx_GiantCorn).BoundingBox)], 'CORN',···
  'Color', 'blue', 'TextBoxOpacity', 1, 'TextColor', 'white');
figure, imshow(Idetect);
```

面積で 2 グループに分け（~ は論理否定の記号），インデックスを付与.

```
idx_Large = ［stats.Area］ > 1500;
idx_Small = ~ idx_Large;
```

– 色による分類小さな面積のグループに対して、色による分類.
各領域の色 (a の値) の平均値を計算，ヒストグラムで確認.

```
stats_a = regionprops(bwNuts, a, 'MeanIntensity');
h = histogram([stats_a(idx_Small).MeanIntensity], 10); ···
  title('a の値')
```

赤と緑で分類し，インデックスを付与.

```
idx_Red= ［stats_a.MeanIntensity］ > 15 & idx_Small;
idx_Green= ［stats_a.MeanIntensity］ < 15 & idx_Small;
Idetect = insertObjectAnnotation(Idetect, 'rectangle',···
 ［cat(1,stats(idx_Red).BoundingBox)］, 'RED', ···
  'Color', 'red', 'TextBoxOpacity', 1);
Idetect = insertObjectAnnotation(Idetect, ···
  'rectangle', ［cat(1,stats(idx_Green).BoundingBox)］,···
  'GREEN', 'Color', 'green', 'TextBoxOpacity', 1);
figure, imshow(Idetect)；確認のため描図.
```

ここまでで 3/5 種類の分類ができた.

– 円形度（Circularity）による分類
大きなグループに対して、円形度での分類を行う.

```
h = histogram([stats(idx_Large & ···
  ~ idx_GiantCorn).Circularity] , 9); title(' 円形度');
```

きれいに分かれていることが確認できるので円形度で分類，インデックスを付与.

```
idx_Almond = [stats.Circularity] > 0.8 & ...
  ~ idx_GiantCorn & idx_Large;
idx_Cashew = [stats.Circularity] < 0.8 & ...
  ~ idx_GiantCorn & idx_Large;
Idetect = insertObjectAnnotation(Idetect,...
  'rectangle',[cat(1,stats(idx_Almond).BoundingBox)],...
  'ALMOND', 'Color', 'magenta', 'TextBoxOpacity', 1);
Idetect = insertObjectAnnotation(Idetect, ...
  'rectangle',[cat(1,stats(idx_Cashew).BoundingBox)], ...
  'CASHEW', 'Color', 'yellow', 'TextBoxOpacity',1);
figure, imshow(Idetect)
```

以上で，形，大きさ，色，円形度，などの 5 種類を分類した.

- 解析：抽出した情報をグループごとに可視化．ラベル行列の作成.
  %以降は書式指定.

```
labels = cell(length(stats), 1);
labels(idx_GiantCorn,1) = {sprintf('%-6s','Corn')};
labels(idx_Red,1) = {sprintf('%-6s','Red')};
labels(idx_Green,1) = {sprintf('%-6s','Green')};
labels(idx_Almond,1) = {sprintf('%-6s','Almond')};
labels(idx_Cashew,1) = {sprintf('%-6s','Cashew')};
labels = cell2mat(labels);
```

データをテーブル型に

```
data_table = struct2table(stats);
label_table = table(labels);
table_all =  [data_table, label_table] ;
```

平均値を表で

```
statarray = grpstats(table_all(:, [1 3:6 end] ), 'labels')
```

表 16.1: データ平均値.

	labels	Group_ Count	mean_ Area	mean_ Eccentricity	mean_ Circularity	mean_Max FereDiam	mean_Min FereDiam
1 Red	Red	15	656.4000	0.8825	0.8242	44.5620	-124.3938
2 Green	Green	8	693.8750	0.8072	0.8393	42.3641	11.5974
3 Almond	Almond	4	2.3623e+03	0.8397	0.8904	75.3938	-67.5016
4 Cashew	Cashew	6	2.4475e+03	0.8093	0.7349	74.0489	-56.9131
5 Corn	Corn	4	3.0768e+03	0.4111	0.9981	67.9186	-92.5418

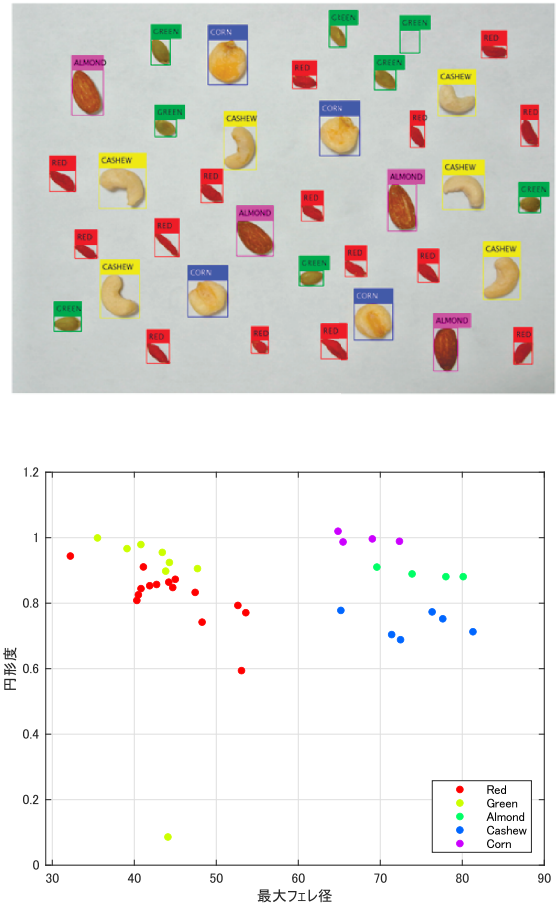


図 16.4: 出力：分類結果．上の図中のラベルの色と下の図のデータ点の色は対応していないことに注意．

グループごとの散布図（図 16.4(下)）

```
figure, gscatter([stats.MaxFeretDiameter], ···
    [stats.Circularity], labels);
xlabel('最大フェレ径'); ylabel('円形度')；
grid on;
```

パラメータごとの相関

```
plotmatrix(table2array(table_all(:, [1 3:6] )))
```

16.3.2　例：ニューラルネットワークによる分類の深層学習（TrainNetwork）

先の例では，画像中のナッツを機械学習で分類しましたが，分類の過程はユーザーがステップごとに指示をしました．次は，手書き数字の認識・分類を（「畳み込みニューラルネットワーク」を用いて）深層学習で行うための，MATLAB が提供するプログラムです．これを用いて説明します．

なお，深層学習についてとりあえず概要を学びたいという人には，
「**ゼロから作る Deep Learning**」斎藤康毅著 オライリー・ジャパン
がお勧めです．これは Python を前提に書かれていますが，Python を知らなくても，全く障害なしに理解することができます．書かれている内容も，特に難しいということはありません．

ここでの深層学習のプロセスは次のような順序で行われます．
1. 入力用画像データの読み込みと確認．
2. ネットワーク アーキテクチャの定義．
3. 学習オプションの指定．
4. ネットワークの学習．
5. 新しいデータのラベルの予測と分類精度の計算．

- **画像データの読み込みと確認**
  `imageDatastore` (イメージ データストア) は，数字の標本データをフォルダー名に基づいて，自動的にラベルを付け，`ImageDatastore` オブジェクトとして格納する．こうすることでデータをバッチ（束）単位で処理できる．

```
digitDatasetPath = fullfile(matlabroot,'toolbox','nnet',...
  'nndemos', 'nndatasets','DigitDataset');
imds = imageDatastore(digitDatasetPath, ...
  'IncludeSubfolders',true,'LabelSource','foldernames');
```

  データストアにある 0〜9 の数字 10000 個の内 20 個のイメージ imds を（4 × 5 の形で）表示．

```
figure;
perm = randperm(10000,20);
for i = 1:20
  subplot(4,5,i);
  imshow(imds.Files{perm(i)});
end
```

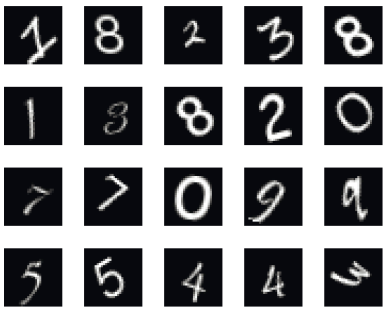


図 16.5: 手書き数字の入力データの例

`labelCount` は，ラベル，およびそれぞれのラベルが付いているイメージの数を格納する table．データストアには 0 〜 9 の数字それぞれについて 1000 個のイメージ，合計で 10000 個のイメージが含まれている．

```
labelCount = countEachLabel(imds)
```

	Label	Count
1	0	1000
2	1	1000
3	2	1000
4	3	1000
5	4	1000
6	5	1000
7	6	1000
8	7	1000
9	8	1000
10	9	1000

ネットワークの入力層にイメージのサイズを指定する必要がある．`digitData` の最初のイメージのサイズを確認（イメージはグレースケールなので各 $28 \times 28 \times 1$ ピクセル）．

```
img = readimage(imds,1);
size(img)
```

ans = 1 × 2
　　28　28

imds を学習セットと検証セットに分割する指定
`splitEachLabel` は，データストアにある `imds` を，750 の学習データセット `imdsTrain` と残り 250 個の検証データセット `imdsValidation` に分割する．

```
numTrainFiles = 750;
[imdsTrain, imdsValidation]=splitEachLabel(imds, ···
  numTrainFiles, 'randomize');
```

- **ネットワーク アーキテクチャの定義**
畳み込みニューラル ネットワークの層の指定をする核心部．

```
layers = [
    imageInputLayer([28 28 1])

    convolution2dLayer(3,8,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,16,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,32,'Padding','same')
    batchNormalizationLayer
    reluLayer

    fullyConnectedLayer(10)
    softmaxLayer
    classificationLayer ];
```

Input Layer (**イメージ入力層**)：イメージ サイズ（$28 \times 28 \times 1$）を指定．1 はチャネル サイズ (グレースケールのため 1).

Convolution Layer （**畳み込み層**）： 深層学習の多くの手法が畳み込みニューラルネットワーク (CNN) を基礎に置いている．ここで「**畳み込み演算**」(「**積和演算**」．積和演算は，画像の平滑化などにも用いられている.）を行う．これは元の関数（行列）と畳み込み関数（行列）（フィルター）との積和をとるもので，最初の引数は `filterSize` といい，その値 3 は畳み込み行列のサイズが 3×3 であることを示す．フィルターを縦横にずらしながら走査するステップサイズ（Stride）のデフォルト値は［1　1］となる．'Padding' とすると行列の端に 0 要素の行および列を加えて畳み込み演算が行われる．
1 ステップ毎のスライドで以下の畳み込み演算を行うと

0	**1**	2	3
3	**0**	1	2
2	3	0	1
1	2	3	0

$*$

0	**1**
2	**0**

$\Rightarrow$

7	2	5
4	7	2
5	4	7

左辺の太字部分の 2×2 行列*2×2 行列の積和計算が矢印右辺の 7 を与える．2 つ目の引数 `numFilters` はフィルターの数（入力の同じ領域に結合するニューロンの数）を示す.

batchNormalizationLayer (**バッチ正規化層**)： 大量のデータ（具体的には配列データ）の**束**（**バッチ batch**）をまとめて処理をする．「**正規化（normalize）**」とは変数を標準化すること，例えば平均値 0，偏差 1 にそろえることをいう．ニューラルネットワークを通じて伝播される活性化と勾配を正規化することにより，ネットワークの学習は簡単な最適化問題になり，学習速度を上げる等の効果がある.

reluLayer (**ReLU 層**) ：ReLU(Rectified Linear Unit) 関数は現在広く用いられている非線形活性化関数.

$$h(x) = \begin{cases} 0 & (x \leq 0) \\ x & (x > 0) \end{cases}$$

maxPooling2dLayer (**最大プーリング層**)：畳み込み層 (と活性化関数) の後で, データの縮小処理を行うことがある（Pooling とは,

プログラムのパフォーマンスを向上させるために，オブジェクトを pool に蓄積しておき必要なときに取り出して利用するという技術）．これにより特徴マップの空間サイズが縮小され、冗長な情報が削除される．データ縮小法の 1 つの方法が最大プーリング．最大プーリング層では，最初の引数 `poolSize` によって指定された矩形領域サイズの最大値を返す．例えばこれが [2,2] ならば，高さ 2，幅 2 の領域内の最大値を返す．

0	**1**	2	3
3	**0**	1	2
2	3	0	1
1	2	3	0

⇒

3	2	3
3	3	2
3	3	3

`dropoutLayer` (**ドロップアウト層**) ：例にはないが，layers={　} の中の適当な場所に dropoutLayer(____,'Name',Name) を挟むことがしばしば行われる．ドロップアウト層は，与えられた確率でランダムに入力要素をゼロに設定する．これにより「過学習」を抑制することができる．

`fullyConnectedLayer` (**全結合層**) ：畳み込み層とダウンサンプリング層の後には，1 つ以上の全結合層を配置する．全結合層はニューロンが前の層のすべてのニューロンに結合している層．この層は，前の層によってイメージ全体で学習されたすべての特徴を組み合わせて，より大きなパターンを特定する．最後の全結合層は，これらの特徴を組み合わせてイメージを分類する (最後の全結合層の `OutputSize` パラメーターは，ターゲット データのクラスの数と等しくなる).

`softmaxLayer` (**ソフトマックス層**) ：最後の全結合層の後にソフトマックス層を作成する．これは各結合層の出力 y_i を，全結合層に関して正規化するもので，使用されるのはソフトマックス活性化関数

$$S(y_i) = \frac{\exp(y_i)}{\sum_j \exp(y_j)}$$

である．

`classificationLayer` (**分類層**) ：最後に分類層を置く．これはソフトマックス活性化関数によって各入力について返された確率を使用して，(交差エントロピー) 誤差を計算し，また出力結果からクラスの数を推測する．ここでいう誤差とは「出力が期待する出力とど

れだけ乖離しているか」の目安で，重みやバイアスなどのパラメータを調整する．

- **学習オプションの指定**
  ネットワーク構造の定義後，学習オプションを指定する．深層学習のネットワークに学習させる場合，学習の進行状況を監視するためのオプションを指定する．

  ここでは，初期学習率 0.01 を指定し， モーメンタム項（過学習を抑制するために入れる項）付き確率的勾配降下法 (SGDM) を用いて，ネットワークに学習させる．学習率とは，勾配降下法においてどのくらい勾配方向に比例した成分を混ぜるかという，ミクシングの割合．最初は小さく混ぜ，全体の変化に応じて割合を調整する．

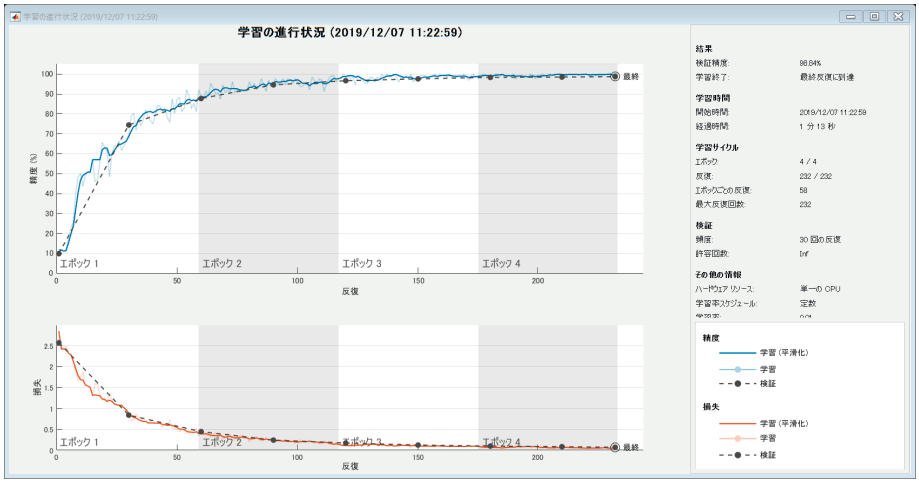
  学習データセット全体の学習サイクル（エポック）の最大回数を 4 に設定．検証データと検証頻度を指定して，学習中にネットワークの精度を監視する．学習データでネットワークに学習させ，学習中に一定間隔で検証データの精度を計算する指定.

```
options = trainingOptions('sgdm','InitialLearnRate',0.01, ...
    'MaxEpochs',4,'Shuffle','every-epoch', ...
    'ValidationData',imdsValidation, ...
    'ValidationFrequency',30, 'Verbose',false,'Plots',...
    'training-progress');
```

- **学習データを使用したネットワークの学習**
  `layers`，学習データおよび学習オプションによって定義されたアーキテクチャを使用して，ネットワークに学習させる．

  学習の進行状況プロットには，ミニバッチの損失と精度および検証の損失と精度が表示される． 深層学習における学習の進行状況の監視を参照.

```
net = trainNetwork(imdsTrain,layers,options);
```



- **検証イメージの分類と精度の計算**
  学習済みネットワークを使用して検証データのラベルを予測し，最終検証精度（ネットワークによって予測が正しく行われるラベルの割合）を計算する．

```
YPred = classify(net,imdsValidation);
YValidation=imdsValidation.Labels;
accuracy=sum(YPred == YValidation)/numel(YValidation)
```

```
accuracy = 0.9888
```

16.3.3　深層学習ネットワーク アーキテクチャの解析（analyzeNetwork ）

ここまでで，深層学習の実際を見てきました．
analyzeNetwork(layers) は，layers によって指定された深層学習ネットワーク アーキテクチャを解析します．

```
analyzeNetwork(layers)
```

関数 analyzeNetwork は、ネットワーク アーキテクチャを対話的に可視化し，各層の詳細情報を提供します．